U.S.P.S. Express Mail Label No.: EV 303 831 354 US

Date of Deposit: October 28, 2003

Attorney Docket No. 15128US02

SYSTEM AND METHOD FOR SECURING DATA

CROSS REFERENCE TO RELATED APPLICATIONS

[01] This application makes reference to, claims priority to and claims benefit from United States Provisional Patent Application Serial No. 60/495,313, entitled "Security Protection for Microprocessor Instructions Using a Single Pipeline Stage Encryption/Decryption Algorithm on a Set-Top-Box (STB) Chip" and filed on August 15, 2003.

INCORPORATION BY REFERENCE

[02] The above-referenced United States patent application is hereby incorporated herein by reference in its entirety.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[03] [Not Applicable]

[MICROFICHE/COPYRIGHT REFERENCE]

[04] [Not Applicable]

BACKGROUND OF THE INVENTION

[05] Integrated circuits may be subject to different types of security attacks. For example, integrated circuits used for decoding and displaying digital television signals such as, for example, set-top box (STB) chips may be hacked by users seeking to access, for free, subscription-services or pay-for-service programming for television. In addition to viewing the programming without paying, successful hackers may be able to distribute "in-the-clear" video content to others.

[06] A hacker may attempt to target different elements of a system. For example, a hacker may target a software component of an STB chip. If software developed by the hacker can

be downloaded onto the STB chip, then on-chip security may be overridden in a variety of ways. In response to such attacks, some STB chips may perform a one-time integrity check on a software code set to ensure, for example, that the software originated from an authorized or recognized source. However, a one-time software integrity check may be open to attacks that may occur after the one-time integrity check has been completed.

[07] Some STB chips may encrypt software instructions in a flash memory or a synchronous dynamic random access memory (SDRAM) memory using data encryption standard (DES) encryption or triple DES (3DES) encryption. By using a DES or 3DES encryption key known only to an authorized entity and the STB, the STB can be assured, with a high degree of confidence, that only microprocessor instructions from an authorized source can be decoded properly. Instruction encryption using DES or 3DES may also protect the code base from examination for clues relating to internal chip operation, even if the flash memory is removed from the board and examined separately. However, DES decryption or 3DES decryption of microprocessor instructions may suffer from, for example, latencies induced in a pipeline to a microprocessor. Pipeline latencies may substantially reduce throughput.

[08] Further limitations and disadvantages of conventional and traditional approaches will become apparent to one of ordinary skill in the art through comparison of such systems with one or more aspects of the present invention as set forth in the remainder of the present application with reference to the drawings.

BRIEF SUMMARY OF THE INVENTION

[09] Aspects of the present invention may be found in, for example, systems and methods that secure data.

[10] In one embodiment, aspects of the present invention may provide a system that protects data. The system may include, for example, a memory and a processor. The memory may store, for example, encrypted data. The processor may be coupled to the memory and may include, for example, a decryptor that decrypts the encrypted data. The decryptor may be adapted, for example, to variably bit roll the encrypted data, to fixedly bit shuffle the bit-rolled data, to add a first key to the bit-shuffled data and to process the added data with a second key.

[11] In another embodiment, aspects of the present invention may provide a system that protects data. The system may include, for example, a memory and a processor. The memory may store, for example, encrypted data. The processor may be coupled to the memory and may include, for example, a decryptor that decrypts the encrypted data without adding a latency to a processor pipeline.

[12] In another embodiment, aspects of the present invention may provide a system that protects data. The system may include, for example, a memory and a processor. The memory may store, for example, encrypted data. The processor may be coupled to the memory and may include, for example, a decryptor that decrypts the encrypted data without adding enough gate delays to exceed a clock cycle budget of the processor.

[13] In another embodiment, aspects of the present invention may provide a system that protects data. The system may include, for example, a memory and a processor. The memory may store, for example, encrypted data. The processor may be coupled to the memory and may include, for example, a decryptor that decrypts the encrypted data and decrypts a word of the encrypted data in a single cycle.

[14] In another embodiment, aspects of the present invention may provide a system that secures data. The system may include, for example, a processor that may decrypt encrypted

data. The processor may be adapted, for example, to variably bit roll encrypted data and to fixedly bit shuffle the bit-rolled data.

[15] In yet another embodiment, aspects of the present invention may provide a method that secures processor instructions. The method may include, for example, one or more of the following: variably rolling data information based on a first key and an address related to the data information; and hard-coded shuffling of the rolled data information; using one or more keys to process the data information.

[16] These and other features and advantages of the present invention may be appreciated from a review of the following detailed description of the present invention, along with the accompanying figures in which like reference numerals refer to like parts throughout.

BRIEF DESCRIPTION OF THE DRAWINGS

[17] FIG. 1 shows a block diagram illustrating an embodiment of a system that protects data according to the present invention.

[18] FIG. 2 shows a flow chart illustrating an embodiment of a method that protects data according to the present invention.

[19] FIG. 3 shows a block diagram illustrating an embodiment of a decryption block according to the present invention.

[20] FIG. 4 shows a block diagram illustrating an embodiment of a portion of a decryption block according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[21] Some embodiments according to the present invention may be related to, for example, systems and methods that protect data. At least one embodiment according to the present invention may be related to, for example, systems and methods that secure computer instructions (e.g., microprocessor instructions). At least one embodiment according to the present invention may be related to, for example, systems and methods that protect software instructions. Some embodiments according to the present invention may employ, for example, a pipeline stage encryption or decryption algorithm (e.g., a single pipeline stage encryption or decryption algorithm). Some embodiments according to the present invention may be used with integrated circuits such as, for example, integrated circuits adapted to decode and to display digital television signals. In at least one embodiment according to the present invention, the integrated circuits may be part of a set top box (STB) or one or more STB chips.

[22] An encryption or decryption scheme (e.g., a single cycle encryption or decryption scheme) for software instructions according to some embodiments of the present invention may include, for example, one or more of the characteristics as set forth below.

[23] The encryption or decryption scheme may add as few gate delays as possible. Fewer gate delays may allow a single cycle encryption or decryption scheme to be merged with existing logic in a pipeline without exceeding a particular clock cycle budget. By keeping within a particular clock cycle budget, additional latencies might not be added to the pipeline.

[24] The encryption or decryption scheme may include, for example, varying elements. For example, some elements may vary periodically such as, for example, at address multiples (e.g., after a particular number of addresses). Periodic variations may enhance security, for example, by changing at least some of the parameters of the encryption or decryption scheme. Furthermore, repeated instructions might not be encoded in the same way each time, thereby thwarting efforts by a hacker to deduce particular portions of the original unencrypted code set.

[25] The encryption or decryption scheme may include, for example, address regions (e.g., multiple address regions) that may be associated with different encryption or decryption algorithms. Encryption or decryption algorithms that change from address region to address region may enhance security, for example, by denying a hacker any insights with respect to an encryption or decryption algorithm by analyzing known instructions in particular address locations such as, for example, jump instructions in a lower address space.

[26] The encryption or decryption scheme may enhance security. For example, some embodiments according to the present invention may be adapted to reduce the risk of deducing an encryption key, a decryption key or an underlying algorithm.

[27] In some embodiments, the present invention may provide a balance between, for example, simplifying an algorithm or logic and obscuring contents (e.g., memory contents) as much as possible. Accordingly, some embodiments according to the present invention may weigh some characteristics in light of other characteristics and may favor some characteristics over other characteristics.

[28] FIG. 1 shows a block diagram illustrating an embodiment of a system that protects data according to the present invention. As illustrated, a processor 100 is coupled to a memory 110 via a bus 120. The processor 100 may include, for example, a decryption block 130 and an internal memory 140. The processor 100 may be, for example, a central processing unit (CPU), a microprocessor or any other computing device. The decryption block 130 may be adapted, for example, to decrypt data from the memory 110. The internal memory 140 may be, for example, an internal cache and may buffer the decrypted data. The memory 110 may be, for example, a flash memory or a random access memory (RAM) such as, for example, a synchronous dynamic RAM (SDRAM). The memory 110 may store, for example, encrypted data such as, for example, encrypted instructions.

[29] The processor 100 or the memory 110 may be part of a set-top box according to some embodiments of the present invention. The processor 100 may be part of a set-top box integrated circuit (e.g., a set-top box chip). The processor 100 and the memory 110 may be

mounted on a circuit board housed by a set-top box. The memory 110 may also be external to or coupled to a set-top box.

[30] In operation according to various embodiments of the present invention, at a head end such as, for example, a service provider, a content provider or a manufacturer may encrypt data before storing the data in the memory 110. Data that has been encrypted by an embodiment of an encryption scheme according to the present invention may be decrypted by an embodiment of a decryption scheme according to the present invention. In some embodiments, additional security measures may be taken. For example, if the memory 110 includes flash memory, but the processor instruction execution is out of SDRAM (e.g., for systems that move the flash memory instructions to SDRAM before beginning execution), then an additional compression step or an encryption step (e.g., a data encryption standard (DES) encryption step or a triple DES (3DES) encryption step) may be performed.

[31] The processor 100 may request (e.g., fetch) a set of data (e.g., a set of instructions) from the memory 110 via the bus 120. Since the data is encrypted, the data may be processed by the decryption block 130, before the data may be stored in the internal cache 140 where the processor 100 may access the data.

[32] FIG. 2 shows a flow chart illustrating an embodiment of a method that decrypts encrypted data according to the present invention. In step 150, the decryption block 130 may perform variable bit rolling of the incoming data bits. The variability of the bit rolling may be related to, for example, a processing of a key and an address. In step 160, the decryption block 130 may perform a fixed bit shuffling of the incoming data bits. The fixed bit shuffling may be, for example, fixed, hard-coded bit shuffling. The hard-coded bit shuffling may differ, for example, for different devices (e.g., different classes of set-top boxes). In step 170, the decryption block 130 may perform processing by adders on the incoming data bits. The adders may be, for example, a series of two-bit adders. A value input into the adders may be related to, for example, a processing of a key and an address. In step 180, the decryption block 130 may perform processing by exclusive OR (XOR) gates. A value input into the XOR gates may be related to, for example, a key (e.g., a hidden internal key).

[33] FIG. 3 shows a block diagram illustrating a decryption block according to the present invention. A decryption block 130 may include, for example, a processor 190, a bit roller 200, a bit swapper 210 (e.g., a fixed bit swapper), an adder 220 and an XOR block 230. The processor 190 may be coupled to the bit roller 200 and to the adder 220. The bit roller 200 may be coupled to the bit swapper 210 which, in turn, may be coupled to the adder 220. The adder 200 may be coupled to the XOR block 230. The bit roller 200 may include, for example, a variable bit roller. The bit swapper 210 may include, for example, a fixed, hard-coded bit shuffler. The adder 220 may include, for example, a set of two-bit adders (e.g., 32 two-bit adders). The XOR block 230 may include, for example, a set of XOR gates.

[34] In operation according to various embodiments of the present invention, a key and an address may be input into the processor 190. Based on, for example, a key and an address, the processor 190 may generate a shifted key and multiplexer selection bits. The shifted key may be sent to the adder 200. The multiplexer selection bits may be sent to the bit roller 200. The bit roller 200 may receive the incoming data, for example, the encrypted data from the memory 110. The bit roller 200 may then perform a bit rolling operation. A bit rolling operation may include, for example, rotating bits within particular roll regions of the incoming data based on, for example, the multiplexer selection bits. The rolled bits may then be sent to the bit swapper 210.

[35] The bit swapper 210 may perform, for example, a fixed bit shuffling (e.g., a fixed, hard-coded bit shuffling) of the bit rolled data. The fixed bit shuffling may differ for different types or classes of chips or devices. The bit swapped data may then be sent to the adder 220. The adder 220 may then combine at least a portion of the shifted key with the bit swapped data using one or more adders (e.g., two-bit adders). The portion of the shifted key used by the adder 220 may change (e.g., shift) as different addresses are input into the processor 190. The output of the adder 220 may be sent to the XOR block 230. The XOR block 230 may then perform an XOR operation with the output of the adder 220 and a key (e.g., an internal key). The key (e.g., a hidden key) may be unrelated to the key used by the processor 190. The key may also be unique for a particular implementation or may be unique for particular versions of chips, for example. The output of the XOR block 230 may

be, for example, decrypted data such as, for example, decrypted instructions for the processor 100. The decrypted data may then be stored in the memory 140 (e.g., an internal cache).

[36] An embodiment of a method for bit rolling according to the present invention is provided. Bit rolling may refer to an ability to rotate bits within a particular roll region within a data set. For example, in a 64-bit data instruction, there may be, for example, six roll regions. Each roll region may be characterized by, for example, a roll skip, a roll region length and a roll amount. These parameters may be set, for example, through bits within at least a portion of a key (e.g., a decryption/encryption key). The bits of the key that may be used to set the parameters may change. For example, the bits of the key that may be used to set the parameters may be based on a data address. The key may be shifted (e.g., periodically, after a particular number of data addresses) so that the roll parameter may be different for different addresses in the memory 110.

[37] For a particular cycle, a mapping of a portion of a key may be set forth as shown below. For example, for two roll regions within a data set having six roll regions, for example, a set of the lower 12 bits of the key, K[11:0], may characterize the roll parameters of the first two roll regions.

Roll Region 1	
K[1:0]	Roll_Skip_1 {00 = start on first bit of Roll Region 1, ..., 11 = start on fourth bit of Roll Region 1}
K[3:2]	Roll_Length_1_5 {00 = roll length of 5, ..., 11 = roll length of 8}
K[5:4]	Roll_Amount_1 {00 = roll amount of 1, ..., 11 = roll amount of 4}
Roll Region 2	
K[7:6]	Roll_Skip_2 {00 = start on first bit of Roll Region 2, ..., 11 = start on fourth bit of Roll Region 2}
K[9:8]	Roll_Length_1_5 {00 = roll length of 5, ..., 11 = roll length of 8}
K[11:10]	Roll_Amount_1 {00 = roll amount of 1, ..., 11 = roll amount of 4}

The mapping may continue up to, for example, six roll regions.

[38] As an example, if the lower 12 bits of the key had a value of 0x476 = 010001110110, then for Roll Region 1 may have the following parameters: Roll_Skip_1 = 10, Roll_Length_1_5 = 01 and Roll_Amount_1 = 11. Thus, in Roll Region 1, the roll may begin

at bit 2 (i.e., the roll may begin at the third bit of Roll Region 1), have a roll region length of six bits and a roll amount of four bits. In the example, Roll Region 2 may have the following parameters: Roll_Skip_2 = 01, Roll_Length_2_5 = 00 and Roll_Amount_2 = 01. Thus, in Roll Region 2, the roll may begin at bit 9 (i.e., the roll starts on the second bit of Roll Region 2), have a roll region length of five bits and a roll amount of two bits.

[39] Thus, the roll parameters might have the following effect on incoming data:

Incoming Data	After Bit Rolling
Bit 0	Unchanged
Bit 1	Unchanged
Bit 2	Bit 6
Bit 3	Bit 7
Bit 4	Bit 2
Bit 5	Bit 3
Bit 6	Bit 4
Bit 7	Bit 5 (End of Roll Region 1)
Bit 8	Unchanged
Bit 9	Bit 11
Bit 10	Bit 12
Bit 11	Bit 13
Bit 12	Bit 9
Bit 13	Bit 10 (End of Roll Region 2)

These roll parameter values may change the next time the key is shifted so that different instructions may have their bits swapped differently.

[40] FIG. 4 shows a block diagram illustrating an embodiment of a portion of a decryption block according to the present invention. FIG. 4 shows some of the components employed in a decryption block and, in particular, some of the components used for decrypting a pair of incoming bits (i.e., D[n] (data_in) and D[n+1] (data_in)). Clearly, the present invention contemplates decrypting more or less than two bits at a time and providing components to decrypt more or less than two bits at a time.

[41] Some of the components of the decryption block may include, for example, the bit roller 200, the bit swapper 210, the adder 220 and the XOR block 230. The bit roller 200 may include, for example, a plurality of multiplexers including multiplexers 240 and 250. The multiplexers 240, 250 may be, for example, 32:1 multiplexers for use with 64-bit

instruction words. Multiplexer selection bits may be input from the processor 190 and may provide the bit rolling as described above. The outputs of the multiplexers may be coupled to the bit swapper 210. The bit swapper 210 may swap bits using, for example, a fixed, hard-coded bit swapper. The output bits of the bit swapper 210 may be sent to the adder 220 which may employ, for example, one or more two-bit adders including XOR gates and AND gates.

[42] As illustrated, the adder 220 includes a two-bit adder. A first output bit of the adder 220 may be a function of an XOR operation between a first output bit of the bit swapper, Dswap[n], and a first key bit of the shifted key, K[n]. The second output bit of the adder 220 may be a function of an XOR operation between (1) an AND operation between K[n] and Dswap[n] and (2) an XOR operation between a second output bit of the bit swapper, Dswap[n+1], and a second key bit of the shifted key, K[n+1].

[43] The output bits of the adder 220 may be sent to the XOR block 230. The XOR block 230 may include, for example, a plurality of XOR gates including XOR gates 300, 310. The XOR gate 300 may perform an XOR operation between a first output bit of the adder 220 and the first bit of a hidden key, HK[n]. The second XOR gate 310 may perform an XOR operation between a second output bit of the adder 220 and the second bit of the a hidden key, HK[n+1]. The output bits of the XOR block 230 may then represent the decrypted data (e.g., original data or instructions).

[44] FIG. 4 also illustrates approximately the number of gate delays per component in the decryption block according to an embodiment of the present invention. For example, the multiplexers 240, 250 may be 32:1 multiplexers with approximately 10 gate delays. The bit swapper 210 may be a fixed, hard-coded bit swapper and thus may have approximately 0 gate delays. The adder 220 may include, for example, a two-bit adder with, at most, approximately 4 gate delays. The XOR block 230 may include, for example, XOR gates 300, 310 with approximately 2 gate delays. Thus, the illustrated arrangement may have between approximately 14 gate delays to approximately 16 gate delays. Some embodiments according to the present invention may have between approximately 14 gate delays to approximately 20 gate delays. Various embodiments according to the present invention

contemplate adjusting configurations or arrangements such that the operation of the decryption block does not exceed a particular clock cycle budget.

[45] Some embodiments according to the present invention generate the multiplexer selection bits and the shifted key in parallel with the main decryption operations and register out the multiplexer selection bits and the shifted key so as not to contribute to the latency in the main pipeline path. In various embodiments according to the present invention, the only gate delays associated with the decryption operation may be the gate delays related to the incoming data as the incoming data passes through the decryption operation. In some embodiments, to process the address in parallel, the decryption operation may take place at least one cycle after the data is fed to the processor 100 from the memory 110.

[46] While the present invention has been described with reference to certain embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted without departing from the scope of the present invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the present invention without departing from its scope. Therefore, it is intended that the present invention not be limited to the particular embodiments disclosed, but that the present invention will include all embodiments falling within the scope of the appended claims.